



**AutoGraFX User Manual**  
Revision 1.1

**Fujitsu Microelectronics America, Inc.**  
September 2008

## Contents

---

Overview.....	3
Development and Build Environment .....	4
Compiler and SDKs:.....	4
Workspace and Project Directory Structure .....	4
Output Directory and Binary Output .....	4
Technical Parameters .....	5
AutoGraFX Common User Interface (AGSCEXTUI.DLL).....	6
AutoGraFX Moniker (AGSCMONIKER.EXE).....	8
AutoGraFX SceneCreator Sink (AGSCSINK.DLL) .....	11
Supporting GIMP as Image Provider.....	12
The GNU Image Manipulation Program (GIMP.EXE).....	12
AutoGraFX SceneCreator GIMP PlugIn (AGSCGIMPPLUGIN.EXE) .....	13

---

## Overview

The Fujitsu AutoGraFX a software solution that addresses the complexity of creating and managing graphical content for Fujitsu GDC applications. The software runs on the Windows 2000/XP/Vista platforms and consists of a Windows application and several specialized components. Native plug-ins for certain imaging applications can be implemented to seamlessly integrate into AutoGraFX and provide the ability to exchange image data in an efficient manner that is conducive to existing workflows for producing image collateral.

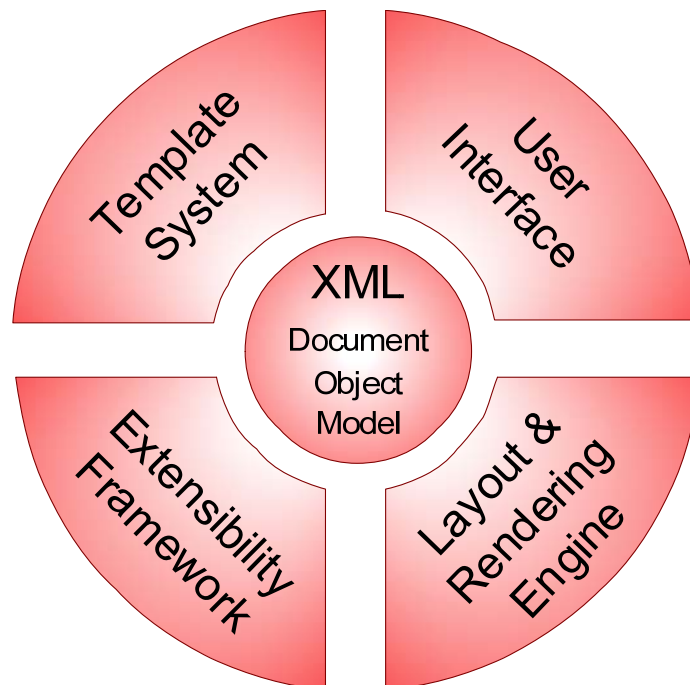


Figure 1.

The lightweight framework described in this document facilitates a mechanism for exchanging image data with external imaging applications. The various components implemented to support imaging applications in a “generic” manner are illustrated in Figure 1 below.

Note! For clarity, the figure does not display additional system DLL dependencies for the various components.

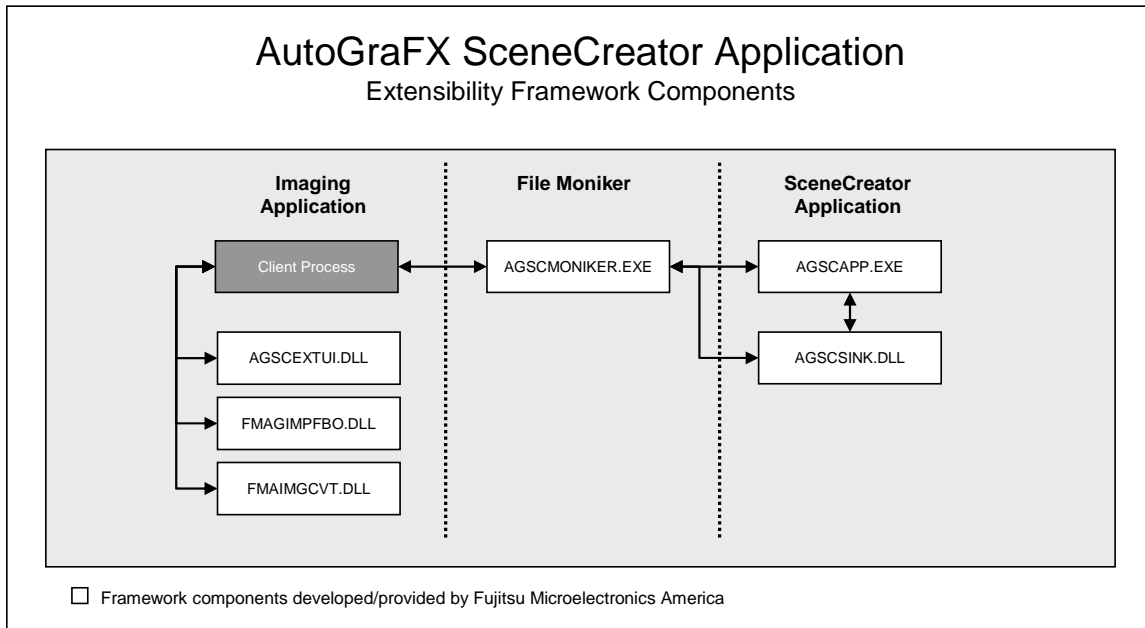


Figure 2.

## Development and Build Environment

### Compiler and SDKs:

The software development environment must include the following:

- Microsoft Visual C++ 2005 with latest service pack.
- Microsoft Platform SDK.
- ATL 8.0
- WTL 8.0

Note! Default installation paths should be used to install the above development components as library and header paths specified in the workspace and custom build steps in projects contained in the workspace currently rely on these paths.

### Workspace and Project Directory Structure

The workspace and the projects contained in it relies on the following directory structures:

```

..\AutoGraFX
..\AutoGraFX\AgScExtUi
..\AutoGraFX\AgScMoniker
..\AutoGraFX\AgScSink
    
```

### Output Directory and Binary Output

```

..\AutoGraFX\Release
..\AutoGraFX\Debug
    
```

## Technical Parameters

The AutoGraFX application provides a light framework to support exchange of image information with external client applications, such as GIMP and Adobe PhotoShop. The framework utilizes the Common Object Model (COM) and consists of three main components: (1) a common user interface module (AGSCEXTUI.DLL), (2) a moniker (AGSCMONIKER.EXE) that connects imaging clients with the AutoGraFX SceneCreator application, and (3) an event sink (AGSCSINK.DLL) used to notify SceneCreator that image data is available for transfer. These components, combined with a native plug-in for the client application, facilitate the inter-process communication required to seamlessly exchange image data between the particular client application and AutoGraFX SceneCreator. The framework also relies on two existing AutoGraFX components, FMAIMGPFBO.DLL and FMAIMGCVT.DLL, to output image data in the Fujitsu Bitmap Object (FBO) format, and to convert image data to and from the various RGB formats used by the applications involved.

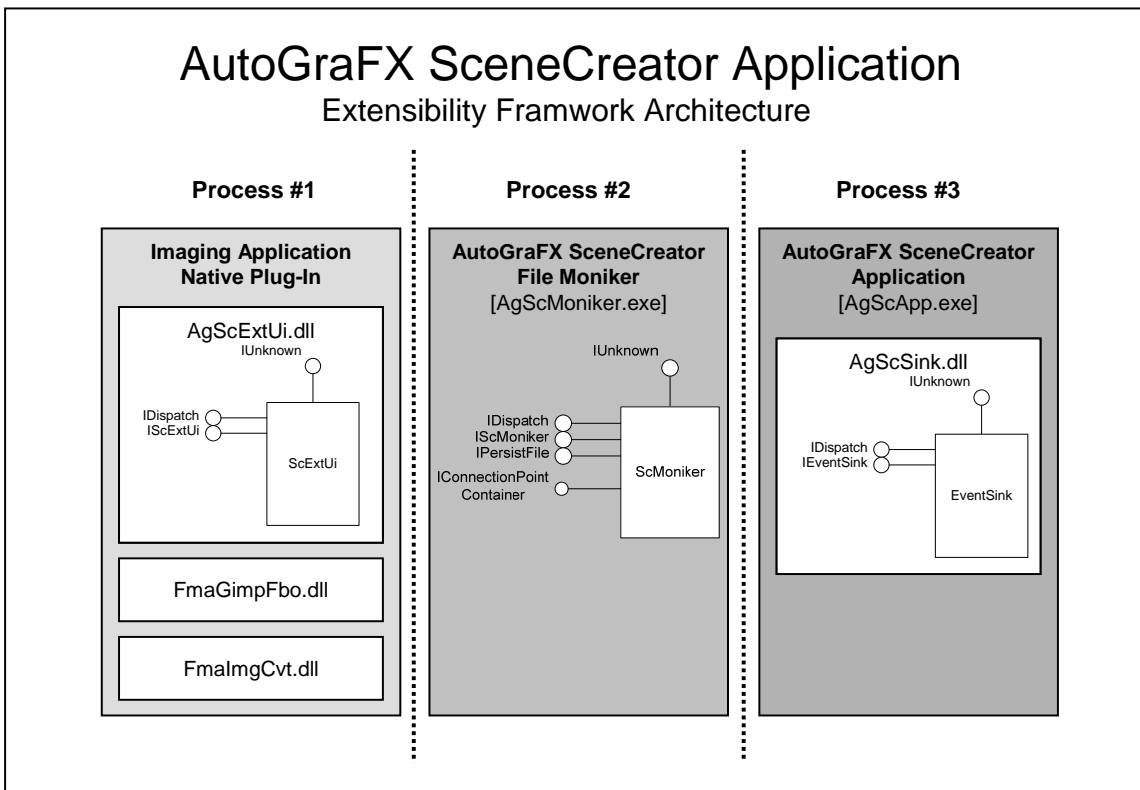


Figure 3.

## AutoGraFX Common User Interface (AGSCEXTUI.DLL)

The AGSCEXTUI.DLL is an in-process COM server that contains the user interface (UI) and functionality for integrating and enabling the transfer of image data from external imaging applications into an existing SceneCreator project. The UI consists of a single dialog box presents UI elements that are intuitive and lets the user select the appropriate properties and actions for transferring image data.

The figure below shows the UI in action:

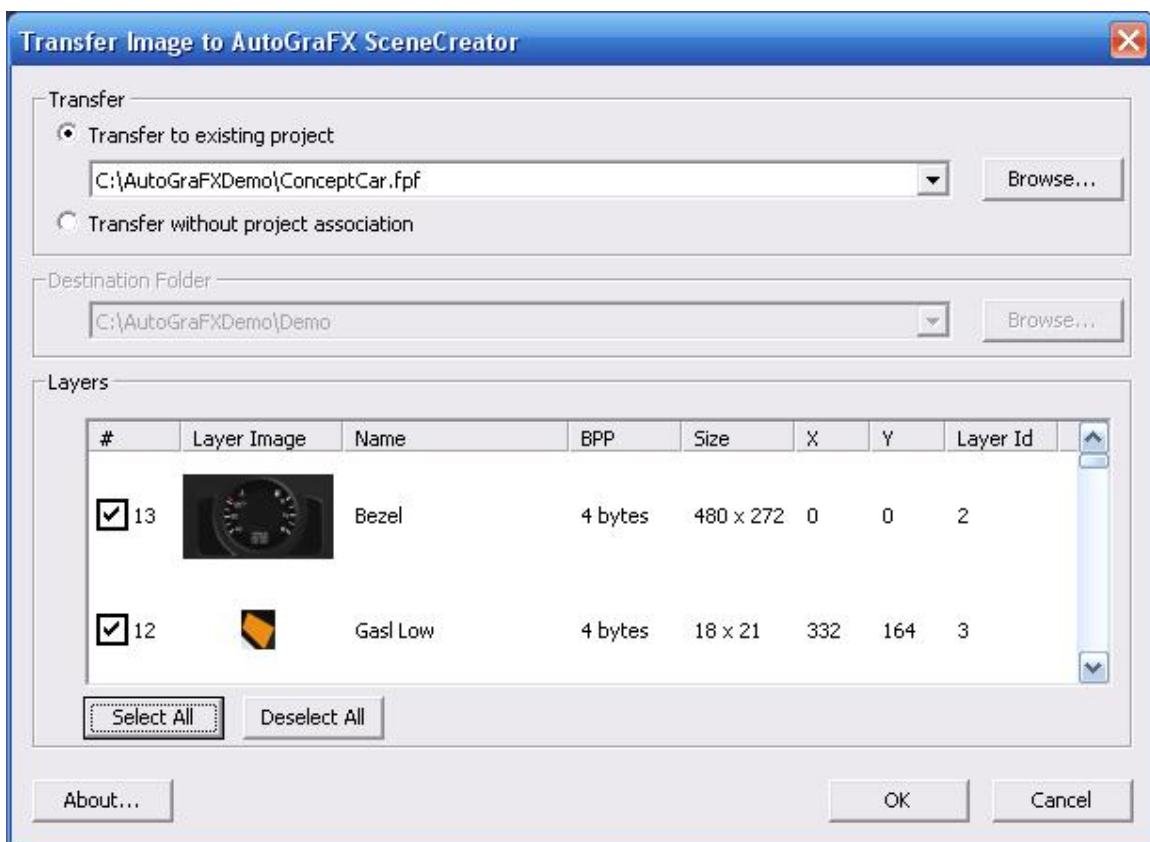


Figure 4.

A client, usually a native plug-in for an external imaging application, can use the IScExtUi interface to select layers from an image composition and obtain the data required to transfer image data to an existing SceneCreator project, or to a specified folder without a SceneCreator project association.

### File-based Transfer

Once the client has obtained information required to transfer image data, it saves the image data related to each layer of the image composition into individual Fujitsu Bitmap Object (FBO) files. The output files are named with the original image name, appended with the layer index from the original image. There are two types of transfers enabled in the UI:

### ***Transfer to an existing project***

When transferring image data to an existing project, the client is responsible for executing the following sequence of operations:

1. Create an instance of the SceneCreator Moniker referencing the specified project.
2. Call the GetSubscriberCount() method on the IScMoniker interface to determine if there are any active instances of the SceneCreator application running with the specified project loaded. If the SceneCreator project is NOT loaded by an existing instance, the client is responsible for launching a new instance of the SceneCreator application with the following command line parameter representing the full qualified path to an existing project:

-File://[FolderPath]\[Filename] (example: -File://C:\AutoFraFXDemo\ConceptCar.fpf)

3. Call the AddFileReference() method on the IScMoniker interface for each of the layers to transfer, providing the path to the FBO file containing the image layer data.
4. Call the FileListReady() method on the IScMoniker interface to indicate that the client has output the image data for each of the specified layers. Calling this method propagates an event to an active instance of the SceneCreator application, with the specified project loaded, through the IEventSink interface. The event is translated into a windows message with the LPARAM parameter containing the number of file references available in a list maintained by the Moniker object, and is sent to the SceneCreator application message loop. The OnFileListReady message handler then calls the GetFileReference() method on the IScMoniker interface to obtain the path to each of the FBO files created. The FBO files are then loaded into the active project as unassigned items.

### ***Transfer without project association***

When transferring image data without association to a project the client is not responsible for any actions beyond saving the image data related to each layer of the image composition into individual Fujitsu Bitmap Object (FBO) files in the specified destination folder.

The SceneCreator Common User Interface Object implements the following interfaces:

#### **Methods in Vtable Order**

<b>IUnknown Methods</b>	<b>Description</b>
QueryInterface	Returns pointers to supported interfaces
AddRef	Increments the reference count
Release	Decrements the reference count

IScExtUi Methods	Description
GetVersion	Gets the version of the User Interface
AddImageLayerDataItem	Adds an image layer data item to the list
RemoveImageLayerDataItem	Removes the specified image layer data item from the list
GetImageLayerDataItemCount	Gets number of image layer data items in the list
DisplayDialog	Displays the user interface
IsImageLayerDataItemSelected	Gets the selection state of the specified image layer data item
GetDestinationFolder	Gets the target folder for the image data output

### AutoGraFX Moniker (AGSCMONIKER.EXE)

The AutoGraFX SceneCreator Moniker contains all the code and data needed to locate the moniker object's COM server and instruct the server to recreate the original object. It is implemented as a simple "File Moniker" that represents the path to a specific AutoGraFX project file. The moniker object is instantiated by calling the Win32 COM API function `CreateFileMoniker()`, which loads the object into the system's Running Object Table (ROT) and provides for multiple clients to bind to the same instance of the object. This enables the moniker to function as a "per-machine-singleton" data transfer object that preserves its state and data independently of client sessions and processes. The following figure shows an active instance of the file moniker registered with the ROT, displayed in the IROTVIEW SDK developer tool application provided by Microsoft:

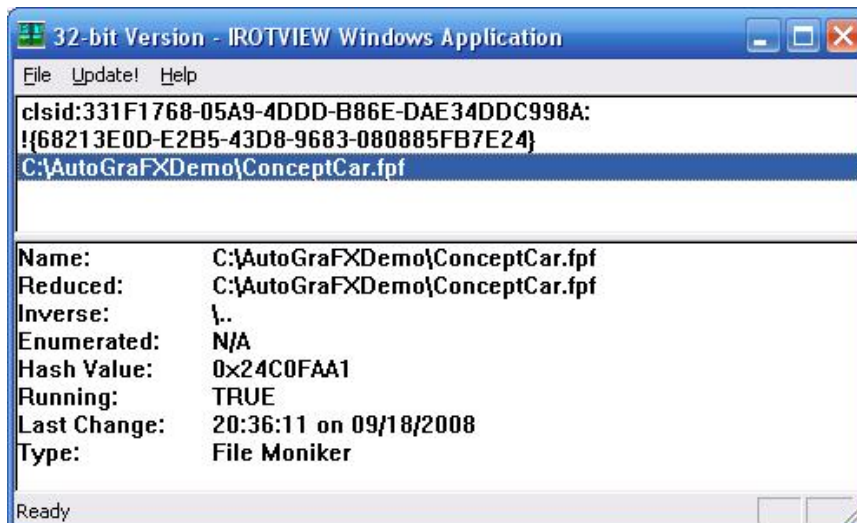


Figure 5.

The File Moniker is a “named” COM object that controls the reference to a particular file that must exist in the file system, and makes an interface pointer to the moniker available to other objects that queries the moniker with the file reference associated with the object. The File Moniker implements the IPersistFile interface so it can be loaded when it is bound. It provides an object-naming scheme only; a naming system that can be used to reference and manipulate an object that is uniquely associated with a Fujitsu Project File, which has the .FPF file extension. The object does not claim exclusive ownership to the referenced file, but implements the IScMoniker interface, which provides a mechanism to control access to the file when clients connect to the interface. The IScMoniker interface contains methods to manage client access, lifetime of the object, event subscription and transfer of image data provided by external image applications. The image data is stored in FBO files in the same folder as the project file referenced by the moniker.

When an instance of the File Moniker is loaded into the ROT, the reference count of the object is incremented with a value greater than the actual reference count to ensure that the object remains in memory. Even when all the clients release their interface pointers to the object, the reference count will be greater than zero. Although this convention breaks the reference counting rules of COM, there is a valid reason for the ROT maintaining a reference count greater than zero; it essentially becomes a caching mechanism. This prevents the object from inadvertently unloading and avoids raise conditions introduced by latencies involved in loading and unloading the object into the ROT. Since the ROT sets a “fake” reference count on the object, the object must implement its own mechanism to “force” unloading without regard for the current reference count. The IScMoniker interface provides the Revoke() method, which the SceneCreator application uses to remove the object from the ROT.

The SceneCreator Moniker Object implements the following interfaces:

#### Methods in Vtable Order

IUnknown Methods	Description
QueryInterface	Returns pointers to supported interfaces
AddRef	Increments the reference count
Release	Decrements the reference count

IPersist Methods	Description
GetClassID	Returns the class identifier (CLSID) for the component object.

<b>IPersistFile Methods</b>	<b>Description</b>
IsDirty	Not Implemented
Load	Opens the specified file and initializes an object from the file contents
Save	Not Implemented
SaveCompleted	Not Implemented
GetCurFile	Not Implemented

<b>IScMoniker Methods</b>	<b>Description</b>
GetVersion	Gets the version of the moniker
GetObjectRefCount	Gets the reference count of the object
Subscribe	Increments event subscription counter value
Unsubscribe	Decrements event subscription counter value
GetSubscriberCount	Gets the number of event subscribers
SetEvent	Increments event counter value
RemoveEvent	Decrements event counter value
GetEventCount	Gets the number of events
AddFileReference	Adds a reference to an FBO file containing image data
GetFileReference	Gets the reference to an FBO file containing image data
GetFileReferenceCount	Gets the number of file references available
RemoveFileReference	Removes a specific file reference
RemoveAllFileReferences	Remove all file references
FileListReady	Notifies clients that file references are available
Revoke	Removes the object from ROT

### AutoGraFX SceneCreator Sink (AGSCSINK.DLL)

The AutoGraFX SceneCreator Event Sink object is an in-process COM server implemented using the classic COM connection point model in which the SceneCreator File Moniker object is the publisher of events and the AutoGraFX SceneCreator Event Sink object is the subscriber. The Event Sink object is created by the SceneCreator application, which first calls the SetEventWindow() method on the interface to set the main application window as a recipient of notifications sent by the sink's event handler. The application registers the subscriber with the publisher by calling the StartListening() method on the IEventSink interface in order to receive events.

The SceneCreator Sink Object implements the following interfaces:

#### Methods in Vtable Order

IUnknown Methods	Description
QueryInterface	Returns pointers to supported interfaces
AddRef	Increments the reference count
Release	Decrements the reference count

IEventSink Methods	Description
StartListening	Registers the event sink with the file moniker and starts listening for events
StopListening	Unregisters the event sink
SetEventWindow	Specifies the window that will receive event notifications

## Supporting GIMP as Image Provider

The GNU Image Manipulation Program (“GIMP”) is a freely distributable software application, commonly used for image authoring, image composition and image retouching.

GIMP’s functionality is extended by implementing a plug-in. Therefore, GIMP requires a native plug-in to extend the new functionality necessary to support the AutoGraFX SceneCreator Extensibility Framework.

The following diagram shows the various components needed to provide support for GIMP as image provider to AutoGraFX SceneCreator:

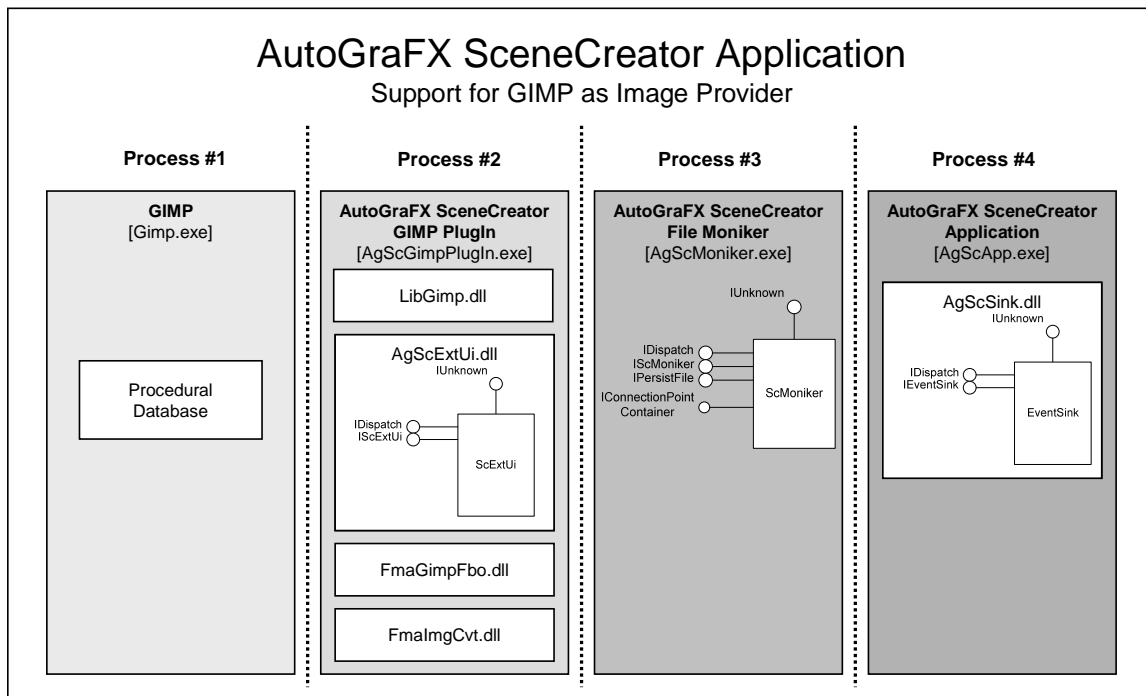


Figure 6.

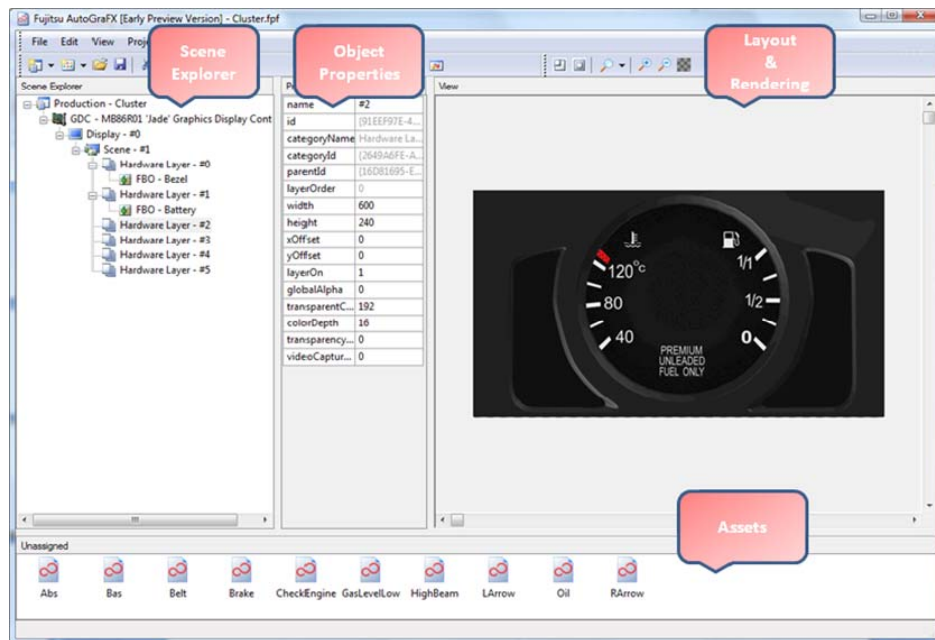
### The GNU Image Manipulation Program (GIMP.EXE)

GIMP uses a Procedural Database (PDB) as an interface to access the functions that make up the core functionality of GIMP. During start-up, GIMP queries all the available plug-ins and enters the function parameters and descriptions reported by the plug-in into the PDB. Essentially, a PDB entry contains all the information required by GIMP when communicating with a plug-in.

## AutoGraFX SceneCreator GIMP PlugIn (AGSCGIMPPLUGIN.EXE)

AutoGraFX SceneCreator GIMP PlugIn is an application that runs in its own dedicated process space. In addition to loading the LIBGIMP library into its memory space, and declaring the various exported functions that the plug-in uses, the plug-in also registers with GIMP and implements certain entry points that are called from the GIMP core utilizing a complex wire-protocol via pipes.

The plug-in communicates with the GIMP core using the various functions available in the LIBGIMP library. These functions provide access to similar functions available in the GIMP core and encapsulate the complexity of the underlying wire protocol used to communicate with the core.



For additional information on AutoGraFX or Fujitsu GDC products, please send your request to the following email address: [Inquiry@fma.fujitsu.com](mailto:Inquiry@fma.fujitsu.com), or visit Fujitsu Microelectronics America, Inc web site at: <http://www.us.fujitsu.com>

FUJITSU MICROELECTRONICS AMERICA, INC.  
 Corporate Headquarters  
 1250 E. Arques Ave. Sunnyvale Ca 94088-3470  
 Ter: (800) 866-8608 Fax: (408) 373-5999  
 E-mail: [Inquiry@fma.fujitsu.com](mailto:Inquiry@fma.fujitsu.com) Web Site: <http://www.us.fujitsu.com>